

Mobile Processor

Project #1

-simple calculator-

Subject: Computer Architecture and Mobile Processor

Professor: Seehwan Yoo

Name: Jueun Park

Student Number: 32191818

date: 2020.03.21

1. Project Introduction

Making a simple calculator that can perform the following operations

- Binary arithmetic operations (+, -, *, /)
- Move operation (move some value to register)
- Storing some value to R0

2. Motivation

The calculator mimics the operation of the CPU. Therefore, it is helpful to understand the operational behavior inside the CPU to implement the calculator.

3. Concepts used in CPU simulation

File operations:

The variable `f` is defined through `'open()'`. After that, enter the file via `'f.write()'`. You can close the file with `'f.close()'`. Prepare to load the file through `'import os'`. The file can be retrieved via `'os.path.exists()'`. The contents of the file can be retrieved and saved through `'f.read()'`.

4. Program Structure

First, write the file `'input.txt'`. Then, call up the expression you want to calculate by reading the file. Expressions imported from a file use a two-dimensional list to separate and store each element. The calculation is performed in a loop and the `'output.txt'` file is written once every run. When the loop is closed, the calculator is also shut down.

5. Problems and solutions

It was difficult for me to distinguish hexadecimal from text at first. When I forced the transformation, I thought about how to distinguish the hexadecimal number from the character because there would be an error if a text came in. To solve this problem, the characteristic that `"0x"` is attached in front of hexadecimal is used. After receiving the value in String, it was divided into hexadecimal numbers if it contained `'0x'` or letters if not.

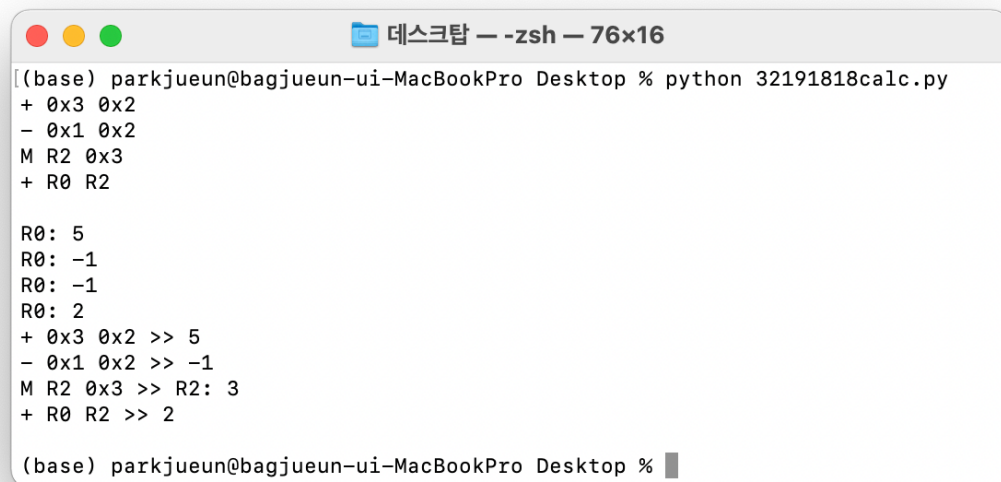
6. Build environment

Compile: Mac OS terminal, with Python

To compile, please type:

python source_code_name.py

7. Screen capture

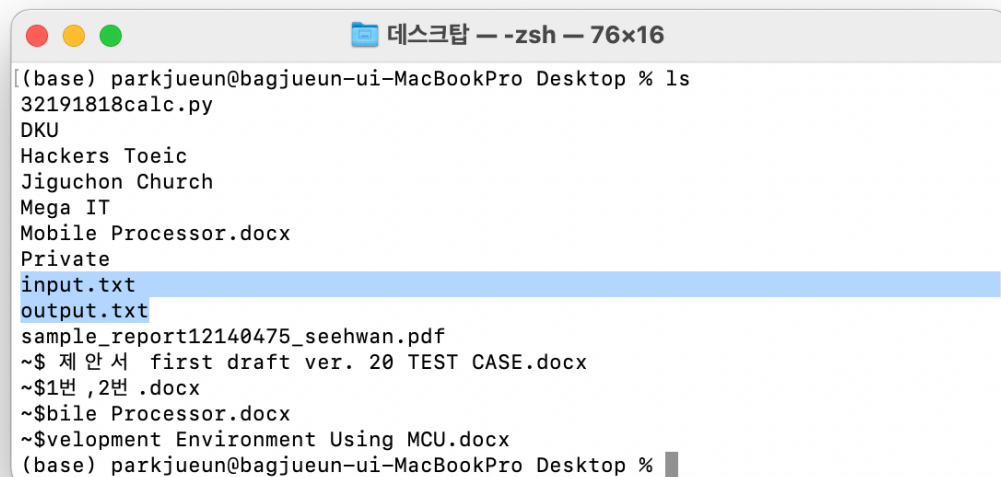


A terminal window titled "데스크탑 — -zsh — 76x16" showing the execution of a Python script. The prompt is "(base) parkjueun@bagjueun-ui-MacBookPro Desktop %". The command executed is "python 32191818calc.py". The output consists of several lines of arithmetic operations and register values: "+ 0x3 0x2", "- 0x1 0x2", "M R2 0x3", "+ R0 R2", "R0: 5", "R0: -1", "R0: -1", "R0: 2", "+ 0x3 0x2 >> 5", "- 0x1 0x2 >> -1", "M R2 0x3 >> R2: 3", and "+ R0 R2 >> 2". The prompt returns to "(base) parkjueun@bagjueun-ui-MacBookPro Desktop %".

```
((base) parkjueun@bagjueun-ui-MacBookPro Desktop % python 32191818calc.py
+ 0x3 0x2
- 0x1 0x2
M R2 0x3
+ R0 R2

R0: 5
R0: -1
R0: -1
R0: 2
+ 0x3 0x2 >> 5
- 0x1 0x2 >> -1
M R2 0x3 >> R2: 3
+ R0 R2 >> 2

(base) parkjueun@bagjueun-ui-MacBookPro Desktop %
```



A terminal window titled "데스크탑 — -zsh — 76x16" showing the output of the "ls" command. The prompt is "(base) parkjueun@bagjueun-ui-MacBookPro Desktop %". The command executed is "ls". The output lists the following files and directories: "32191818calc.py", "DKU", "Hackers Toeic", "Jiguchon Church", "Mega IT", "Mobile Processor.docx", "Private", "input.txt", "output.txt", "sample_report12140475_seehwan.pdf", "~\$ 제안서 first draft ver. 20 TEST CASE.docx", "~\$1번, 2번 .docx", "~\$bile Processor.docx", and "~\$velopment Environment Using MCU.docx". The prompt returns to "(base) parkjueun@bagjueun-ui-MacBookPro Desktop %".

```
((base) parkjueun@bagjueun-ui-MacBookPro Desktop % ls
32191818calc.py
DKU
Hackers Toeic
Jiguchon Church
Mega IT
Mobile Processor.docx
Private
input.txt
output.txt
sample_report12140475_seehwan.pdf
~$ 제안서 first draft ver. 20 TEST CASE.docx
~$1번, 2번 .docx
~$bile Processor.docx
~$velopment Environment Using MCU.docx
(base) parkjueun@bagjueun-ui-MacBookPro Desktop %
```

